

# Codes With Hierarchical Locality

Birenjith Sasidharan, Gaurav Kumar Agarwal, and P. Vijay Kumar

Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore.

Email: biren, agarwal, vijay@ece.iisc.ernet.in

## Abstract

In this paper, we study the notion of *codes with hierarchical locality* that is identified as another approach to local recovery from multiple erasures. The well-known class of *codes with locality* is said to possess hierarchical locality with a single level. In a *code with two-level hierarchical locality*, every symbol is protected by an inner-most local code, and another middle-level code of larger dimension containing the local code. We first consider codes with two levels of hierarchical locality, derive an upper bound on the minimum distance, and provide optimal code constructions of low field-size under certain parameter sets. Subsequently, we generalize both the bound and the constructions to hierarchical locality of arbitrary levels.

## Index Terms

Codes with locality, locally recoverable codes, hierarchical locality, multiple erasures, distributed storage.

## I. INTRODUCTION

An important desirable attribute in a distributed storage system is the efficiency in carrying out repair of failed nodes. Among many others, two important metrics to characterize efficiency of node repair are *repair bandwidth*, i.e., the amount of data download in the case of a node failure and *repair degree*, i.e., the number of helper nodes accessed for node repair. While regenerating codes [1] aim to minimize the repair bandwidth, codes with locality [2] seek to minimize the repair degree. The focus of the present paper is on codes with locality.

### A. Codes with Locality

An  $[n, k, d]$  linear code  $\mathcal{C}$  can possibly require to access  $k$  symbols to recover one lost symbol. The notion of locality of code symbols was introduced in [2], with the aim of designing codes in such a way that the number of symbols accessed to repair a lost symbol is much smaller than the dimension  $k$  of the code. The code  $\mathcal{C}$  is said to have locality  $r$  if the  $i$ -th code symbol  $c_i$ ,  $1 \leq i \leq n$  can be recovered by accessing  $r \ll k$  other code symbols. In [2], authors proved an upper bound on the minimum distance of codes with locality, and showed that an existing family of pyramid codes [3] can achieve the bound. In [4], authors extended the notion to  $(r, \delta)$ -locality, where each symbol can be recovered *locally* even in the presence of an additional  $(\delta - 2)$  erasures. In [2], authors introduced categories of *information-symbol* and *all-symbol* locality. In the former, local recoverability is guaranteed for symbols from an information set, while in the latter, it is guaranteed for every symbol. Explicit constructions for codes with all-symbol locality are provided in [5], [6], respectively based on rank-distance and Reed-Solomon (RS) codes. Improved bounds on the minimum distance of codes with all-symbol locality are provided in [7], [8], along with certain optimal constructions. Families of codes with all-symbol locality with small alphabet size (low field size) are constructed in [9]. Locally repairable codes over binary alphabet are constructed in [10]. A new approach of *local regeneration*, where in repair is both local and in addition bandwidth-efficient within the local group, achievable by making use of a vector alphabet is considered in [4], [11], [12].

Recently, many approaches are proposed in literature [4], [7], [9], [13], [14] to address the problem of recovering from multiple erasures locally. The notion of  $(r, \delta)$ -locality introduced in [4] is one such. In [13], an approach of protecting a single symbol by multiple support-disjoint local codes of the same length is considered. An upper bound on the minimum distance is derived, and existence of optimal codes is established under certain constraints. A similar approach is considered in [9] also. In [9], authors allow *multiple recovering sets* of different sizes, and also provide constructions requiring field-size only in the order of block-length. Quite differently, authors of [7] consider codes allowing sequential recovery of two erasures, motivated by the fact that such a family of codes allow a larger minimum distance. An upper bound on the minimum distance and optimal constructions for restricted set of parameters are provided.

### B. Our Contributions

In the present paper, we study the notion of *hierarchical locality* that is identified as another approach to local recovery from multiple erasures. In consideration of practical distributed storage systems, Duminuco et al. in [15] had proposed the topology

This research is supported in part by the National Science Foundation under Grant No. 1422955 and in part by the joint UGC-ISF Research Grant No. 1676/14. Birenjith Sasidharan would like to acknowledge the support of TCS Research Scholar Programme Fellowship.

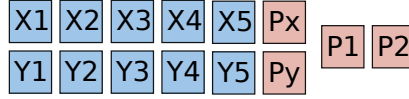


Fig. 1: Illustration of  $[16, 12, 4]$ -code used in Windows Azure.

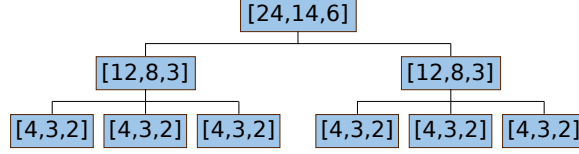


Fig. 2: Illustration of  $[24, 14, 6]$ -code having 2-level hierarchical locality.

of *hierarchical codes* earlier. They compared hierarchical codes with RS codes in terms of repair-efficiency using real network-traces of KAD and PlanetLab networks. Their work was focused on collecting empirical data for performance improvements, rather than undertaking a theoretical study of such a topology. In the present paper, we study codes with hierarchical locality, first considering the case of two-level hierarchy. We derive an upper bound on the minimum distance and provide optimal code constructions under certain parameter-sets. This is further generalized to a setting of  $h$ -*hierarchy* in a straightforward manner.

## II. CODES WITH HIERARCHICAL LOCALITY

The Windows Azure Storage solution employs a  $[16, 12, 4]$ -pyramid code with a locality parameter  $r = 6$ . In the code, as illustrated in Fig. 1, every code symbol except the global parities  $P_1, P_2$  can be recovered accessing  $r = 6$  other code symbols. While the code performs well in systems where single node-failure remains the dominant event, it requires to connect to  $k = 12$  symbols to recover a failed under certain erasure-patterns consisting of 2 node-failures. We consider an example of  $[24, 14, 6]$ -code from the family of *codes with hierarchical locality* in an attempt to reduce such an overhead. The structure of the code is depicted in Fig. 2 as a tree in which each node represents a constituent code. The code contains two support-disjoint  $[n_1 = 12, r_1 = 8, d_2 = 3]$  codes, each of them in turn comprised of three support-disjoint  $[n_2 = 4, r_2 = 3, d_1 = 2]$  codes. Making use of  $[4, 3, 2]$ -code, all single-erasures can be repaired accessing  $r_2 = 3$  symbols, which is half the number of symbols required in the Windows Azure code in a similar situation. We can recover a lost symbol connecting to  $r_1 = 8$  symbols in the case of erasure-pattern involving 2 erasures. This is in contrast to the Windows Azure code where we had to download the entire message of 12 symbols. While the Windows Azure code offers a storage overhead of 1.3x with a minimum distance of 4, our code has a larger overhead of 1.7x with a better minimum distance  $d = 6$ . The example of  $[24, 14, 6]$ -code can indeed be constructed, and it will be shown that the minimum distance is optimal among the class of codes.

### A. Preliminaries

**Definition 1:** [4] An  $[n, k, d]$  linear code  $\mathcal{C}$  is a code with  $(r, \delta)$ -locality if for every symbol  $c_i, 1 \leq i \leq n$ , there exists a punctured code  $C_i$  such that  $c_i \in \text{Supp}(C_i)$  and the following conditions hold: 1)  $\dim(C_i) \leq r$ , 2)  $d_{\min}(C_i) \geq \delta$ .

Codes with locality were first defined in [2] for the case of  $\delta = 2$ , and the class was generalized for arbitrary  $\delta$  in [4]. In the definition given in [4], the authors imposed constraints on the length and the  $d_{\min}$  of  $C_i$ . We replace the constraint on length with a constraint on  $\dim(C_i)$ , and it may be noted that it does not introduce any loss in generality. The code  $C_i$  associated with the  $i$ -th symbol is referred to as its *local code*. If it is sufficient to have local codes only for symbols belonging to some fixed information set  $I$ , such codes are referred to as *codes with information-symbol  $(r, \delta)$ -locality*. The general class in Def. 1 is also referred to as *codes with all-symbol  $(r, \delta)$ -locality*, in order to differentiate them from the former. In this paper, unless otherwise mentioned, we consider codes with all-symbol locality.

**Definition 2:** An  $[n, k, d]$  linear code  $\mathcal{C}$  is a *code with hierarchical locality* having locality parameters  $[(r_1, \delta_1), (r_2, \delta_2)]$  if for every symbol  $c_i, 1 \leq i \leq n$ , there exists a punctured code  $C_i$  such that  $c_i \in \text{Supp}(C_i)$  and the following conditions hold: 1)  $\dim(C_i) \leq r_1$ , 2)  $d_{\min}(C_i) \geq \delta_1$ , 3)  $C_i$  is a code with  $(r_2, \delta_2)$ -locality.

The punctured code  $C_i$  associated with  $c_i$  is referred to as its *middle code*. Since the middle code is a code with locality, each of its symbols will in turn be associated with a local code.

### B. An Upper Bound On the Minimum Distance

**Theorem 2.1:** Let  $\mathcal{C}$  be an  $[n, k, d]$ -linear code with hierarchical locality having locality parameters  $[(r_2, \delta_2), (r_1, \delta_1)]$ . Then

$$d \leq n - k + 1 - \left( \left\lceil \frac{k}{r_2} \right\rceil - 1 \right) (\delta_2 - 1) - \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) (\delta_1 - \delta_2). \quad (1)$$

*Proof:* We extend the techniques introduced in [2] in proving the theorem. A punctured code  $\mathcal{C}_s$  of  $\mathcal{C}$  having dimension  $k - 1$ , is identified first. Then we will use the fact that

$$d \leq n - |\text{Supp}(\mathcal{C}_s)|. \quad (2)$$

The Algorithm 1 (see flow chart in App. A) is used to find  $\mathcal{C}_s$  with a large support. In each iteration indexed by  $j$ , the algorithm identifies a middle code from  $\mathcal{C}$ , that accumulates additional rank. Then it picks up local codes from within the middle code that accumulate additional rank. Clearly, the algorithm terminates as the total rank is bounded by  $k$ . Let  $i_{\text{end}}$  and  $j_{\text{end}}$  respectively denote the final values of the variables  $i$  and  $j$  before the algorithm terminates. Let  $a_i$  denote the incremental rank and  $s_i$  denote the incremental support while adding a local code  $L_i$ . Then we have  $s_i \geq a_i + (\delta_2 - 1)$ ,  $1 \leq i \leq i_{\text{end}}$ , since we have  $a_i > 0$  in every iteration. The set  $S_i$  denotes the support of  $L_i$ , and  $V_i$  denotes the space  $\text{Column-Space}(G|_{S_i})$ , where  $G$  is the generator matrix of the code. If no more local codes can be added from the middle code  $M_j$ , then the support of the last local code added from  $M_j$  is removed and an additional support  $T_j$  of  $M_j$  is added to  $\Psi$ . Let  $i(j)$  denote the index of the last local code added from  $M_j$ . Since the middle code has a minimum distance of  $\delta_1$ , and every rank accumulating local code brings at least one new information symbol, it follows that

$$\begin{aligned} t_j := |T_j| &\geq a_{i(j)} + (\delta_1 - 1), \\ &= a_{i(j)} + (\delta_2 - 1) + (\delta_1 - \delta_2) \quad 1 \leq j \leq j_{\text{end}}. \end{aligned}$$

---

**Algorithm 1** For the proof of Thm. 2.1

---

```

1: Let  $j = 0, i = 0, W = \phi, \Psi = \phi$ .
2: while  $(\exists \text{ a middle code } M_j \in \mathcal{C} \text{ such that } \text{rank}(G|_{\Psi \cup M_j}) > \text{rank}(G|_{\Psi}))$  do
3:   while  $(\exists \text{ a local code } L_i \in M_j \text{ such that } V_i \subsetneq W)$  do
4:      $W = W + V_i$ 
5:      $\Psi = \Psi \cup S_i$ 
6:      $i = i + 1$ 
7:   end while
8:    $\Psi = (\Psi \setminus S_{i-1}) \cup T_j$ 
9:    $j = j + 1$ 
10: end while

```

---

The rank accumulates to  $k$  after adding the last local code  $L_{i_{\text{end}}}$ . We would also have visited  $j_{\text{end}}$  middle codes by then. Hence,

$$i_{\text{end}} \geq \left\lceil \frac{k}{r_2} \right\rceil, \quad j_{\text{end}} \geq \left\lceil \frac{k}{r_1} \right\rceil. \quad (3)$$

After adding  $L_{i_{\text{end}}-1}$  local codes, we would have accumulated rank that is less than or equal to  $(k-1)$ . Hence we can always pick  $s_e := (k-1) - \sum_{i=1}^{i_{\text{end}}-1} a_i$  columns from  $L_{i_{\text{end}}}$  so that the total rank accumulated becomes  $(k-1)$ . Note that  $s_e \geq 0$ . The resultant punctured code is identified as  $\mathcal{C}_s$ . Let  $E = \{i(j) \mid 1 \leq j \leq j_{\text{end}}\}$ . Then

$$|\text{Supp}(\mathcal{C}_s)| \geq \sum_{i \notin E, i=1}^{i_{\text{end}}-1} s_i + s_e + \sum_{j=1}^{j_{\text{end}}-1} t_j. \quad (4)$$

In (4), the last term  $\sum_{j=1}^{j_{\text{end}}-1} t_j$  includes a sum of only  $j_{\text{end}} - 1$  terms because we could have possibly accumulated a rank of  $(k-1)$  after adding  $L_{i_{\text{end}}-1}$ , i.e.,  $s_e = 0$ . Thus we have,

$$\begin{aligned} |\text{Supp}(\mathcal{C}_s)| &\geq \sum_{i \notin E, i=1}^{i_{\text{end}}-1} s_i + (k-1) - \sum_{i=1}^{i_{\text{end}}-1} a_i + \sum_{j=1}^{j_{\text{end}}-1} t_j \\ &\geq \sum_{i \notin E, i=1}^{i_{\text{end}}-1} (a_i + (\delta_2 - 1)) + (k-1) - \sum_{i=1}^{i_{\text{end}}-1} a_i \\ &\quad + \sum_{j=1}^{j_{\text{end}}-1} (a_{i(j)} + (\delta_2 - 1) + (\delta_1 - \delta_2)) \\ &= \sum_{i=1}^{i_{\text{end}}-1} (\delta_2 - 1) + (k-1) + \sum_{j=1}^{j_{\text{end}}-1} (\delta_1 - \delta_2) \end{aligned}$$

Substituting values of  $i_{\text{end}}$  and  $j_{\text{end}}$  from (3) and using (2), we obtain the bound. ■

It may be noted that the theorem holds good even for codes with information-symbol hierarchical locality.

### C. Code Constructions For Information-Symbol Locality

A straightforward extension of pyramid codes [3] is possible to construct optimal codes with information-symbol hierarchical locality. They achieve the bound in (1) if  $r_2 \mid r_1 \mid k$ . In this section, we illustrate the construction with an example, assuming  $\delta_2 = 2$ . The two-level hierarchical code described here extends naturally to multiple-level hierarchy and yields optimal codes if  $r_h \mid r_{h-1} \mid \dots \mid r_1 \mid k$ .

The construction is built on a systematic MDS code with parameters  $[k + d - 1, k, d]$  with a generator matrix  $G_{\text{mds}}$ . Let

$$\begin{aligned} G_{\text{mds}} &= [I_{k \times k} \mid Q_{k \times (d-1)}] \\ k &= \alpha r_1 + \beta r_2 + \gamma, \quad 0 \leq \beta r_2 + \gamma < r_1 \\ r_1 &= \mu r_2 + \nu, \quad 0 \leq \nu < r_2. \end{aligned}$$

Partition  $Q$  as

$$Q = \left[ \begin{array}{c|c} Q_1 \\ Q_2 \\ \vdots \\ Q_{\alpha+1} \end{array} \middle| Q' \right],$$

where  $Q_i, 1 \leq i \leq \alpha$  is of size  $r_1 \times (\delta_1 - 1)$ ,  $Q_{\alpha+1}$  is of size  $(\beta r_2 + \gamma) \times (\delta_1 - 1)$  and  $Q'$  is of size  $(k \times (d - \delta_1))$ . Further partition  $Q_i, 1 \leq i \leq \alpha$  as

$$Q_i = \left[ \begin{array}{c|c} R_{i1} \\ R_{i2} \\ \vdots \\ R_{i,\mu+1} \end{array} \middle| R'_i \right],$$

where  $R_{ij}, 1 \leq j \leq \mu$  is of size  $r_2 \times 1$ ,  $R_{i,\mu+1}$  is of size  $\nu \times 1$  and  $R'_i$  is of size  $(r_1 \times (\delta_1 - 2))$ . At the same time  $Q_{\alpha+1}$  is partitioned as

$$Q_{\alpha+1} = \left[ \begin{array}{c|c} R_{\alpha+1,1} \\ R_{\alpha+1,2} \\ \vdots \\ R_{\alpha+1,\beta+1} \end{array} \middle| R'_{\alpha+1} \right],$$

where  $R_{\alpha+1,j}, 1 \leq j \leq \beta$  is of size  $r_2 \times 1$ ,  $R_{\alpha+1,\beta+1}$  is of size  $\gamma \times 1$  and  $R'_{\alpha+1}$  is of size  $((\beta r_2 + \gamma) \times (\delta_1 - 2))$ . Next, we can construct matrices

$$\begin{aligned} \hat{Q}_i &= \left[ \begin{array}{cccc|c} R_{i1} & & & & \\ & R_{i2} & & & \\ & & \ddots & & \\ & & & R_{i,\mu+1} & \\ & & & & R'_i \end{array} \right] \\ \hat{Q}_{\alpha+1} &= \left[ \begin{array}{cccc|c} R_{\alpha+1,1} & & & & \\ & R_{\alpha+1,2} & & & \\ & & \ddots & & \\ & & & R_{\alpha+1,\beta+1} & \\ & & & & R'_{\alpha+1} \end{array} \right] \\ \hat{Q} &= \left[ \begin{array}{cccc|c} \hat{Q}_1 & & & & \\ & \hat{Q}_2 & & & \\ & & \ddots & & \\ & & & \hat{Q}_{\alpha+1} & \\ & & & & Q' \end{array} \right] \end{aligned}$$

Let  $J$  be defined as

$$J = \left[ \begin{array}{cccc} I_{r_2} & & & \\ & \ddots & & \\ & & I_{r_2} & \\ & & & I_\nu \end{array} \right].$$

where  $I_{r_2}$  is repeated  $\mu$  times. Finally we construct the generator matrix  $G$  of the pyramid code as,

$$G = \left[ \begin{array}{cccccc|c} J & & & & & & \\ & \ddots & & & & & \\ & & J & & & & \\ & & & I_{r_2} & & & \\ & & & & \ddots & & \\ & & & & & I_{r_2} & \\ & & & & & & I_{r_2} & \\ & & & & & & & I_{r_2} \end{array} \right] \hat{Q},$$

with  $J$  being repeated  $\alpha$  times, and  $I_{r_2}$  repeated  $\beta$  times. The resultant code has length

$$n = k + d - 1 + \left( \left\lceil \frac{k}{r_1} \right\rceil \left\lceil \frac{r_1}{r_2} \right\rceil - 1 \right) + \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) (\delta_1 - 2).$$

Clearly, minimum distance of the code corresponding to  $G$  is greater than or equal to that of  $G_{\text{mds}}$ , which is  $d$ . Also,  $G$  satisfies the property of information-symbol hierarchical locality by construction. Using the bound in (1), the code is optimal if

$$\left\lceil \frac{k}{r_1} \right\rceil \left\lceil \frac{r_1}{r_2} \right\rceil = \left\lceil \frac{k}{r_2} \right\rceil.$$

#### D. Code Constructions For All-Symbol Locality

We assume a divisibility condition  $n_2 \mid n_1 \mid n$ . The construction is described in three parts. The first part involves identification of a suitable finite field  $\mathbb{F}_{p^m}$ , a partition of  $\mathbb{F}_{p^m}^*$  and a set of polynomials in  $\mathbb{F}_{p^m}[X]$  that satisfy certain conditions. We require that every polynomial evaluates to a constant within one subset in the partition, and evaluates to zero in all the remaining subsets. In the second part, we construct a code polynomial  $c(X)$  from the message symbols with the aid of the suitably chosen polynomials. The code polynomial  $c(X)$  is formed in such a way that the locality constraints are satisfied. This part also involves precoding of message symbols in such a way that the dimension of the middle codes and the global code are kept to the desired values. Finally, the third part involves evaluation of the code polynomial at  $n$  points of  $\mathbb{F}_{p^m}^*$ , that are chosen in the first part.

1) *Identification of  $\mathbb{F}_{p^m}$ , a partition of  $\mathbb{F}_{p^m}^*$  and a set of polynomials:* Let the finite field  $\mathbb{F}_{p^m}$  be such that  $n_1 \mid p^m - 1$ , and  $n < p^m$ . Existence of such a pair  $(p, m)$  is shown in App. B. We define the integers

$$n_0 = p^m - 1, \mu_0 = 1, \mu_1 = \frac{n_0}{n_1}, \mu_2 = \frac{n_1}{n_2}.$$

Let  $\alpha$  denote the primitive element of  $\mathbb{F}_{p^m}$ , and hence  $\mathbb{F}_{p^m}^* = \{1, \alpha, \alpha^2, \dots, \alpha^{p^m-2}\}$ . Set  $\beta_0 = \alpha$  and  $\beta_1, \beta_2$  be elements of order  $n_1$  and  $n_2$  respectively. Then we have the following subgroups:

$$\begin{aligned} H_0 &= \mathbb{F}_{p^m}^* \\ H_1 &= \{1, \beta_1, \beta_1^2, \dots, \beta_1^{n_1-1}\} \\ H_2 &= \{1, \beta_2, \beta_2^2, \dots, \beta_2^{n_2-1}\}. \end{aligned}$$

We can further write

$$\begin{aligned} H_0 &= H_1 \uplus \beta_0 H_1 \uplus \dots \uplus \beta_0^{\mu_1-1} H_1 \\ H_1 &= H_2 \uplus \beta_1 H_2 \uplus \dots \uplus \beta_1^{\mu_2-1} H_2. \end{aligned}$$

Having set up a subgroup chain, we proceed to define a family of subsets of  $H_0$ . These subsets are indexed by a tuple  $(i, \underline{t})$  with  $i \in \{0, 1, 2\}$ . For a given value of  $i$ ,  $\underline{t}$  takes values from the set  $T_i = \{(t_0, t_1, t_2) \mid 1 \leq t_j \leq \mu_j \text{ for } j \leq i; t_j = 0 \text{ for } j > i\}$ . For a given tuple  $(i, \underline{t})$ , let us define a coset  $A_{(i, \underline{t})}$  of the subgroup  $H_i$  as follows:

$$\gamma_{(i, \underline{t})} = \prod_{j=0}^{i-1} \beta_j^{t_{j+1}-1}, \quad A_{(i, \underline{t})} = \gamma_{(i, \underline{t})} H_i.$$

The set of possible indices has a tree-structure with each index  $(i, \underline{t})$  associated with a unique vertex of the tree. A vertex  $(i, \underline{t})$  belongs to the  $i$ -th level of the tree, and the 3-tuple  $\underline{t}$  describes the unique path from the vertex to the root of the tree. The parent of a vertex  $(i, \underline{t})$  is denoted by  $\pi(i, \underline{t})$ , and the set of its siblings, i.e., other vertices having the same parent, is denoted by  $\psi(i, \underline{t})$ . The tree structure of  $\{A_{(i, \underline{t})} \mid i \in \{0, 1, 2\}, \underline{t} \in T_i\}$  is depicted in Fig. 3. Since each vertex at  $i$ -th level is associated with a coset of  $H_i$ , we refer to this tree as the *coset-tree*.

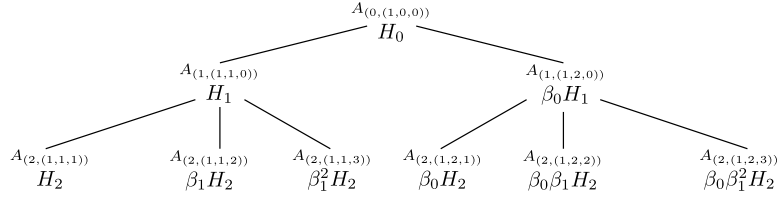


Fig. 3: Illustration of the coset-tree.

Next, we define the polynomials  $p_{(i,\underline{t})}(X), q_{(i,\underline{t})}(X) \in \mathbb{F}_{p^m}[X]$  as,

$$p_{(i,\underline{t})}(X) = \prod_{\theta \in A_{(i,\underline{t})}} (X - \theta), \quad q_{(i,\underline{t})}(X) = \prod_{\underline{s} \in \psi(i,\underline{t})} p_{(i,\underline{s})}(X).$$

The polynomial  $p_{(i,\underline{t})}(X)$  is the annihilator of  $A_{(i,\underline{t})}$ . Furthermore, the polynomials in  $\{q_{(i,\underline{t})}(X) \mid 1 \leq t_i \leq \mu_i, t_j \text{ is fixed for } j \neq i\}$  are relatively prime collectively. Thus for  $i = 1, 2$  there exists  $\{a_{(i,\underline{t})}(X) \mid 1 \leq t_i \leq \mu_i, t_j \text{ is fixed for } j \neq i\}$  such that

$$\sum_{\substack{\underline{s} \\ t_i=1 \\ t_j \text{ fixed for } j \neq i}}^{\mu_i} a_{(i,\underline{s})}(X) q_{(i,\underline{t})}(X) = 1 \bmod (X^{n_{i-1}} - \gamma_{(i-1,\underline{s})}^{n_{i-1}}), \quad (5)$$

where  $\underline{s} = \pi(i, \underline{t})$ . Next, we define  $E_{(i,\underline{t})}(X) = a_{(i,\underline{t})}(X) q_{(i,\underline{t})}(X)$ , and determine a valid candidate for  $E_{(i,\underline{t})}(X)$  in the next Lemma such that (5) holds. Subsequently in Lem. 2.3, we will list down certain useful properties of these polynomials. The proof is relegated to Appendix.

*Lemma 2.2:*

$$E_{(i,\underline{t})}(X) = \prod_{\underline{s} \in \psi(i,\underline{t})} \left[ \frac{X^{n_i} - (\gamma_{(i,\underline{s})})^{n_i}}{(\gamma_{(i,\underline{t})})^{n_i} - (\gamma_{(i,\underline{s})})^{n_i}} \right]. \quad (6)$$

*Lemma 2.3:* Let  $i \in \{1, 2\}$ ,  $t, \underline{s} \in T_i$  and  $\underline{t} \in \psi(i, \underline{s})$ . Let  $\underline{\tau} = (\tau_0, \tau_1, \tau_2) = \pi(i, \underline{t})$ . Then

$$E_{(i,\underline{t})}(X) = g(X^{n_i}) \text{ for some polynomial } g(\cdot), \deg(g) = \mu_i - 1 \quad (7)$$

$$E_{(i,\underline{t})}(\theta) = \begin{cases} 1 & \theta \in A_{(i,\underline{t})}(X) \\ 0 & \theta \in \gamma_{(i-1,\underline{\tau})} H_{i-1} \setminus A_{(i,\underline{t})} \end{cases} \quad (8)$$

$$E_{(i,\underline{s})}(X) E_{(i,\underline{t})}(X) \equiv 0 \bmod (X^{n_{i-1}} - \gamma_{(i-1,\underline{\tau})}^{n_{i-1}}) \quad (9)$$

$$E_{(i,\underline{t})}^2(X) = E_{(i,\underline{t})}(X) \bmod (X^{n_{i-1}} - \gamma_{(i-1,\underline{\tau})}^{n_{i-1}}). \quad (10)$$

*Proof:* The property (7) is clear from the definition of  $E_{(i,\underline{t})}(X)$ . The properties (8), (9) are clear from the proof of Lemma 2.2. Hence (10) follows by (5).  $\blacksquare$

2) *Construction of  $c(X)$ :* We start with associating message polynomials of degree  $(r_2 - 1)$  with certain leaves of the coset-tree. The total number of leaves of the coset-tree equals  $\mu_1 \mu_2$ . However, we will only consider a suitable subtree of the coset-tree such that the number of leaves equals  $\bar{\mu}_1 \mu_2$  where  $\bar{\mu}_1 = \frac{n}{n_1}$ . The required subtree is obtained by removing the last  $(\mu_1 - \bar{\mu}_1)$  branches emanating from the root of the tree. Every leaf that is retained in the subtree has an index  $(2, \underline{t})$  where  $\underline{t}$  belongs to the set

$$T'_2 = \{\underline{t} \in T_2 \mid 1 \leq t_1 \leq \bar{\mu}_1\}.$$

This subtree is referred to as the *relevant coset-tree*. A vertex from the  $i$ -th level,  $i > 0$  of the relevant coset-tree will have an index  $(i, \underline{t})$  where  $\underline{t} \in T'_i = \{\underline{t} \in T_i \mid 1 \leq t_1 \leq \bar{\mu}_1\}$ .

Consider a set  $\mathcal{U} = \{u_{\underline{t}}(X) = u_{\underline{t},0} + u_{\underline{t},1}X + \dots + u_{\underline{t},r_2-1}X^{r_2-1} \mid \underline{t} \in T'_2\}$  of message polynomials of size  $\bar{\mu}_1 \mu_2$ . The code polynomial  $c(X)$  is built from  $\mathcal{U}$  in an iterative manner. In every iteration, we take as input a set of polynomials corresponding to vertices of the  $i$ -th level of the relevant coset-tree, and output another set of polynomials corresponding to vertices of the  $(i - 1)$ -th level. As noted earlier, each leaf of the relevant coset-tree is uniquely mapped to a polynomial in  $\mathcal{U}$ . In the end, we will identify a polynomial  $c_{(0,(1,0,0))}(X)$  associated with the root of the relevant coset-tree. The code polynomial  $c(X) = c_{(0,(1,0,0))}(X)$ . It may be noted that the polynomials in  $\mathcal{U}$  is made up of  $\bar{\mu}_1 \mu_2 r_2$  message symbols in total. However, the desired dimension  $k$  can be less than  $\bar{\mu}_1 \mu_2 r_2$ . Hence in every iteration, a precoding of message symbols is carried out causing a reduction in the number of independent message symbols. The dimension would be reduced to the desired value  $k$  at the end of the final iteration.

Let us now start the iteration by setting  $c_{(2,\underline{t})}(X) = u_{\underline{t}}(X) \ \forall \underline{t} \in T'_2$ . Evaluations of  $c_{(2,\underline{t})}(X)$  at  $n_2$  points in  $A_{(2,\underline{t})}(X)$  give rise to an  $[n_2, r_2]$ -codeword. Recognizing this correspondence, we refer to  $c_{(2,\underline{t})}(X), \underline{t} \in T'_2$  as a *second level code polynomial*. In the next iteration, for every  $\underline{t} \in T'_1$

$$d_{(1,\underline{t})}(X) = \sum_{\underline{s}: \pi(2,\underline{s})=(1,\underline{t})} c_{(2,\underline{s})}(X) E_{(2,\underline{s})}(X).$$

By (7) in Lemma 2.3, the coefficient of  $X^\ell$  is zero in  $E_{(2,\underline{t})}(X)$  whenever  $\ell \neq 0 \pmod{n_2}$ . Hence for every  $\underline{t} \in T'_1$ , there are  $\mu_2 r_2$  monomials in  $d_{(1,\underline{t})}(X)$ . Evaluations of  $d_{(1,\underline{t})}(X)$  at  $n_1$  points in  $A_{(1,\underline{t})}(X)$  give rise to an  $[n_1, \mu_2 r_2]$ -codeword. Since the desired dimension of the middle code is  $r_1$ , we precode the message symbols such that the coefficients of  $(r_2 \mu_2 - r_1)$  highest degree monomials in  $d_{(1,\underline{t})}(X)$  vanishes to zero. The polynomials  $c_{(1,\underline{t})}(X)$  thus obtained corresponds to an  $[n_1, r_1]$ -middle code, and hence referred to as a *first level code polynomial*. We can write

$$c_{(1,\underline{t})}(X) = \sum_{\underline{s}: \pi(2,\underline{s})=(1,\underline{t})} P_1(c_{(2,\underline{s})}(X)) E_{(2,\underline{s})}(X), \quad \underline{t} \in T'_1,$$

where  $P_1(\cdot)$  denotes the precoding transformation at the first level. In the next iteration, we compute  $d_{(0,(1,0,0))}(X)$  and subsequently precode the message symbols by  $P_0(\cdot)$  to reduce the dimension from  $\bar{\mu}_1 r_1$  to  $k$  to obtain the *zeroth level code polynomial*  $c_{(0,(1,0,0))}(X)$ :

$$\begin{aligned} d_{(0,(1,0,0))}(X) &= \sum_{\substack{\underline{s}: \pi(1,\underline{s})=(0,(1,0,0)) \\ \underline{s} \in T'_1}} c_{(1,\underline{s})}(X) E_{(1,\underline{s})}(X) \\ c_{(0,(1,0,0))}(X) &= \sum_{\substack{\underline{s}: \pi(1,\underline{s})=(0,(1,0,0)) \\ \underline{s} \in T'_1}} P_0(c_{(1,\underline{s})}(X)) E_{(1,\underline{s})}(X), \end{aligned}$$

The code polynomial  $c(X)$  is identified as

$$c(X) = c_{(0,(1,0,0))}(X). \quad (11)$$

3) *Evaluation of  $c(X)$* : The codeword  $\underline{c} = (c(\theta) \mid \theta \in A)$  is obtained by evaluating the polynomial  $c(X)$  at  $n$  points taken from

$$A = \bigcup_{\underline{t} \in T'_1} A_{(1,\underline{t})}.$$

This completes the description of the construction. By the construction, it is clear that the dimension and the minimum distance of the code are given by

$$\begin{aligned} k &= |\{\ell \mid \text{coefficient of } X^\ell \text{ in } c(X) \neq 0\}| \\ d &\geq n - \deg(c(X)). \end{aligned}$$

*Remark 1*: A principal construction in [9] for codes with all-symbol locality, relies on a partitioning of the roots of unity contained in a finite field into a subgroup and its cosets. The construction then identifies polynomials that are constant on each coset and makes use of these polynomials in the construction. The approach adopted here is along similar lines.

*Example 1*: In this example, we construct a code with  $[n, k] = [24, 14]$  having locality parameters  $(n_1, r_1) = (12, 8)$  and  $(n_2, r_2) = (4, 3)$ , satisfying the divisibility condition. We can choose the finite field  $\mathbb{F}_{p^m} = \mathbb{F}_{5^2}$ . Let  $\alpha$  be a primitive element of  $\mathbb{F}_{5^2}$ . We have  $n_0 = n = 24$ ,  $\mu_1 = \bar{\mu}_1 = 2$ , and  $\mu_2 = 3$ . We set

$$\begin{aligned} H_0 &= \mathbb{F}_{5^2}^* \\ H_1 &= \{1, \beta_1, \beta_1^2, \dots, \beta_1^{11}\} \\ H_2 &= \{1, \beta_2, \beta_2^2, \beta_2^3, \dots, \beta_2^{11}\} \end{aligned}$$

where  $\beta_0 = \alpha$ ,  $\beta_1 = \alpha^2$  and  $\beta_2 = \alpha^6$ . The relevant coset-tree can be computed as

$$\begin{aligned} A_{(0,(1,0,0))} &= H_0, \quad A_{(1,(1,1,0))} = H_1, \quad A_{(1,(1,2,0))} = \beta_0 H_1 \\ A_{(2,(1,t_1,t_2))} &= \beta_0^{t_1-1} \beta_1^{t_2-1} H_2, \quad 1 \leq t_1 \leq 2, 1 \leq t_2 \leq 3. \end{aligned}$$

Let us define the index sets  $T_1 = \{(t_0, t_1, t_2) \mid t_0 = 1, 1 \leq t_1 \leq 2, t_2 = 0\}$ ,  $T_2 = \{(t_0, t_1, t_2) \mid t_0 = 1, 1 \leq t_1 \leq 2, 1 \leq t_2 \leq 3\}$ . For every  $\underline{t} = (1, t_1, 0) \in T_1$ , we set  $s_1$  as the unique element in  $\{1, 2\} \setminus \{t_1\}$  and then we have

$$\begin{aligned} E_{(1,\underline{t})}(X) &= \left( \frac{X^{12} - (\beta_0^{s_1-1})^{12}}{(\beta_0^{t_1-1})^{12} - (\beta_0^{s_1-1})^{12}} \right) \\ &:= a_{\underline{t}} X^{12} + b_{\underline{t}}. \end{aligned}$$

Similarly for every  $\underline{t} = (1, t_1, t_2) \in T_2$ , we set  $\{s_1, s_2\} = \{1, 2, 3\} \setminus \{t_2\}$  and then we have

$$\begin{aligned} E_{(2,\underline{t})}(X) &= \left( \frac{X^4 - (\beta_0^{t_1-1} \beta_1^{s_1-1})^4}{(\beta_0^{t_1-1} \beta_1^{t_2-1})^4 - (\beta_0^{t_1} \beta_1^{s_1-1})^4} \right) \left( \frac{X^4 - (\beta_0^{t_1-1} \beta_1^{s_2-1})^4}{(\beta_0^{t_1-1} \beta_1^{t_2-1})^4 - (\beta_0^{t_1} \beta_1^{s_2-1})^4} \right) \\ &:= e_{\underline{t}} X^8 + f_{\underline{t}} X^4 + g_{\underline{t}}. \end{aligned}$$

There are  $|T_2| = 6$  message polynomials denoted by  $\{u_{\underline{t}}(X) = u_{\underline{t},0} + u_{\underline{t},1}X + u_{\underline{t},2}X^2 \mid \underline{t} \in T_2\}$ , each of degree  $(r_2 - 1) = 2$ . The second level code polynomial for each  $\underline{t} \in T_2$  corresponding to a  $[4, 3]$ -local code is taken to be  $c_{(2,\underline{t})}(X) = u_{\underline{t}}(X)$ . In the next step, the first level code polynomial  $\{c_{1,\underline{s}}(X)\}$  is constructed as

$$c_{1,\underline{t}}(X) = \sum_{\underline{s}: s_1=t_1, s_2=1}^{s_2=2} P_1(c_{(2,\underline{s})}(X))(e_{\underline{s}}X^8 + f_{\underline{s}}X^4 + g_{\underline{s}}).$$

for each of  $\underline{t} \in T_1$ . By virtue of the precoding  $P_1(\cdot)$ , the term  $X^{10}$  vanishes and the resultant polynomial  $c_{1,\underline{t}}(X)$  corresponds to a  $[12, 8]$ -middle code. Subsequently, the zeroth level code polynomial is constructed as

$$c_{0,(1,0,0)}(X) = \sum_{\underline{s}: s_1=1, s_2=0}^{s_1=2} P_0(c_{(1,\underline{s})}(X))(a_{\underline{s}}X^{12} + b_{\underline{s}}).$$

Without precoding  $P_0(\cdot)$ , we would have obtained a polynomial of degree 21 having 16 monomials. Precoding wipes out the terms  $\{X^{21}, X^{20}\}$ , and the resultant polynomial  $c_{0,(1,0,0)}(X) =: c(X)$  of degree 18 is the code polynomial consisting of 14 monomials. Thus  $k = 14$ , and  $d \geq 6$ . The codeword  $\underline{c}$  is given by  $\underline{c} = (c(\theta) \mid \theta \in H_0)$ .

It is of interest to look at the exponents of monomials in polynomials of each level. From each level, we pick a candidate polynomial  $c(X)$ ,  $c_1(X) := c_{(1,(1,1,0))}(X)$ ,  $c_2(X) := c_{(2,(1,1,2))}(X)$ .

<del>3</del>	2	1	0
--------------	---	---	---

<del>11</del>	<del>10</del>	9	8	<del>7</del>	6	5	4	<del>3</del>	2	1	0
---------------	---------------	---	---	--------------	---	---	---	--------------	---	---	---

<del>23</del>	<del>22</del>	<del>21</del>	<del>20</del>	<del>19</del>	18	17	16	<del>15</del>	14	13	12
<del>11</del>	<del>10</del>	9	8	<del>7</del>	6	5	4	<del>3</del>	2	1	0

Fig. 4: Illustration of the exponents of monomials in  $c_2(X)$ ,  $c_1(X)$  and  $c(X)$  in order. The canceled exponents are those whose coefficients are fixed to zero by precoding.

The illustration in Fig. 4 gives an equivalent simplistic description of the  $[24, 14, 6]$  code. This works in general. Let  $\text{Exp}(f)$  represent the ordered set of exponents of the monomials in a polynomial  $f(X)$ . By ordered set, we mean that the elements of the set are listed in the descending order. For example,  $\text{Exp}(X^3 + \alpha X + 1) = \{3, 1, 0\}$ . For an ordered finite set  $S$  of non-negative integers and a positive integer  $r \leq |S|$ , we define  $\text{Trunc}(S, r)$  as the set comprising of the last  $r$  elements of the set. Then we have that

$$\begin{aligned} \text{Exp}(c_2) &= \{r_2 - 1, r_2 - 2, \dots, 0\} = \text{Trunc}(\mathbb{Z}_{n_2}, r_2) \\ \text{Exp}(c_1) &= \text{Trunc}\left(\bigcup_{j=0}^{\mu_2-1} (jn_2 + \text{Exp}(c_2)), r_1\right) \\ \text{Exp}(c) &= \text{Trunc}\left(\bigcup_{j=0}^{\bar{\mu}_1-1} (jn_1 + \text{Exp}(c_1)), k\right), \end{aligned}$$

where  $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$ . The set  $\text{Exp}(c)$  is an equivalent simplistic description of the code. In terms of  $\text{Exp}(c)$ , we can write the parameters of the code as  $k = |\text{Exp}(c)|$ ,  $d \geq n - \max(\text{Exp}(c))$ .

### E. Locality Properties Of the Code

In this section, we will show that the code satisfies locality constraints. Consider the case  $c(y)$  is lost. We need to recover it accessing  $r_1$  other symbols  $\{c(y_1), c(y_2), \dots, c(y_{r_1})\}$  that along with  $c(y)$  are part of an  $[n_1, r_1]$  punctured code. Without loss of generality, let us assume that  $y \in A_{(1,(1,1,0))}$ . Using (9), (10) in Lemma 2.3, we can write

$$c(X)E_{(1,(1,1,0))}(X) = P_0(c_{(1,(1,1,0))}(X))E_{(1,(1,1,0))}(X).$$

Evaluations at  $r_1$  out of  $n_1$  points in  $A_{(1,(1,1,0))}$  will help reconstruct  $P_0(c_{(1,(1,1,0))}(X))$ , since  $\deg(P_0(c_{(1,(1,1,0))}(X))) \leq (r_1 - 1)$ . Then we can recover  $c(y) = P_0(c_{(1,(1,1,0))}(y))E_{(1,(1,1,0))}(y)$ . The same argument can be used inductively to show that each symbol within an  $[n_1, r_1]$ -middle code can be recovered by  $r_2$  out of some  $n_2$  symbols. This establishes the existence of  $[n_2, r_2]$ -local codes.



### F. Optimality Of the Code

**Theorem 2.4:** The  $[n, k, d]$ -code with  $[n_1, r_1, \delta_1]$ -middle codes and  $[n_2, r_2, \delta_2]$ -local codes constructed in Sec. II-D achieves the optimal minimum distance if  $r_2 \mid r_1 \mid k$ .

*Proof:* Let  $c(X)$  denote the code polynomial. The proof follows from counting  $\mathbb{Z}_n$  in two different ways. We can write

$$n = |\mathbb{Z}_n| = |\text{Exp}(c)| + |\mathbb{Z}_n \setminus \text{Exp}(c)|. \quad (12)$$

We have that  $|\text{Exp}(c)| = k$ . On the other hand, since  $r_2 \mid r_1 \mid k$ , we can count the number of exponents that are truncated while forming  $c$  as,

$$\begin{aligned} |\mathbb{Z}_n \setminus \text{Exp}(c)| &= \left( \frac{k}{r_2} - 1 \right) (\delta_2 - 1) \\ &\quad + \left( \frac{k}{r_1} - 1 \right) (\delta_1 - \delta_2) + (d - 1) \end{aligned}$$

Substituting back in (12), we conclude that the code is optimal.  $\blacksquare$

**Theorem 2.5:** The  $[n, k, d]$ -code with  $[n_1, r_1, \delta_1]$ -middle codes and  $[n_2, r_2, \delta_2]$ -local codes constructed in Sec. II-D achieves the optimal minimum distance if the following conditions hold:

- 1)  $d = n_2 + \delta_2$
- 2)  $\frac{n}{n_1} = \left\lceil \frac{k}{r_1} \right\rceil$ ,  $\frac{n}{n_2} = \left\lceil \frac{k}{r_2} \right\rceil + 1$

*Proof:* Let  $c(X)$  denote the code polynomial. The proof is analogous to that of Thm. 2.4. The only difference lies in the count of  $|\mathbb{Z}_n \setminus \text{Exp}(c)|$ . Since  $d = n_2 + \delta_2$ , we obtain that,

$$\begin{aligned} |\mathbb{Z}_n \setminus \text{Exp}(c)| &= \left( \frac{n}{n_2} - 1 \right) (\delta_2 - 1) + \left( \frac{n}{n_1} - 1 \right) (\delta_1 - \delta_2) \\ &\quad + ((d - 1) - (\delta_2 - 1)) \\ &= \left( \frac{n}{n_2} - 2 \right) (\delta_2 - 1) + \left( \frac{n}{n_1} - 1 \right) (\delta_1 - \delta_2) + (d - 1). \end{aligned}$$

Hence the code is optimal if the second condition in the theorem holds.  $\blacksquare$

While Thm. 2.4, Thm. 2.5 provide optimality conditions that can be generalized to hierarchical locality of arbitrary levels, a subject of discussion in Sec. III, the next theorem characterizes the conditions for optimality for two-level hierarchy without imposing any restrictions.

**Theorem 2.6:** The  $[n, k, d]$ -code with  $[n_1, r_1, \delta_1]$ -middle codes and  $[n_2, r_2, \delta_2]$ -local codes constructed in Sec. II-D achieves the optimal minimum distance if

$$\left\lceil \frac{k}{r_2} - \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) \frac{r_1}{r_2} \right\rceil = \left\lceil \frac{k}{r_2} \right\rceil - \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) \left\lceil \frac{r_1}{r_2} \right\rceil. \quad (13)$$

*Proof:* Let  $c(X)$  denote the code polynomial. The proof is analogous to that of Thm. 2.4. It is possible to count the size of  $\mathbb{Z}_n \setminus \text{Exp}(c)$  as

$$\begin{aligned} &\left( \left( \left\lceil \frac{r_1}{r_2} \right\rceil - 1 \right) (\delta_2 - 1) + (\delta_1 - 1) \right) \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) \\ &\quad + \left( \left\lceil \frac{k - \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) r_1}{r_2} \right\rceil - 1 \right) (\delta_2 - 1) + (d - 1). \end{aligned}$$

The expression can be recast into the form

$$\begin{aligned} &\left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) (\delta_1 - \delta_2) + (d - 1) + \\ &\quad \left( \left\lceil \frac{k}{r_2} - \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) \frac{r_1}{r_2} \right\rceil + \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) \left\lceil \frac{r_1}{r_2} \right\rceil - 1 \right) (\delta_2 - 1), \end{aligned}$$

thus leading to a value of  $d$  as in

$$\begin{aligned} d &= n - k + 1 - \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) (\delta_1 - \delta_2) \\ &\quad - \left( \left\lceil \frac{k}{r_2} - \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) \frac{r_1}{r_2} \right\rceil + \left( \left\lceil \frac{k}{r_1} \right\rceil - 1 \right) \left\lceil \frac{r_1}{r_2} \right\rceil - 1 \right) (\delta_2 - 1). \end{aligned}$$

Comparing against the upper bound in (1), we conclude that the code is optimal if (13) holds.  $\blacksquare$

### III. GENERALIZATION TO $h$ -LEVEL HIERARCHY

In Sec. II, we considered codes with hierarchical locality where the hierarchy had two levels. Here we extend the notion to  $h$ -level hierarchy where  $h$  is an arbitrary number.

**Definition 3:** An  $[n, k, d]$  linear code  $\mathcal{C}$  is a *code with  $h$ -level hierarchical locality* having locality parameters  $[(r_1, \delta_1), (r_2, \delta_2), \dots, (r_h, \delta_h)]$  if for every symbol  $c_i, 1 \leq i \leq n$ , there exists a punctured code  $C_i$  such that  $c_i \in \text{Supp}(C_i)$  and the following conditions hold:

- 1)  $\dim(C_i) \leq r_1$ ,
- 2)  $d_{\min}(C_i) \geq \delta_1$ ,
- 3)  $C_i$  is a code with  $(h-1)$ -level hierarchical locality having locality parameters  $[(r_2, \delta_2), (r_3, \delta_3), \dots, (r_h, \delta_h)]$ .

Code with 1-level hierarchical locality is defined to be code with locality.

The punctured code  $C_i$  associated with  $c_i$  is referred to as *local code of level-1*. In fact, each symbol is associated with a bunch of local codes, each of level- $i, i = 1, 2, \dots, h$ . In the previous section, we studied codes with 2-level hierarchical locality.

#### A. An Upper Bound on The Minimum Distance

**Theorem 3.1:** Let  $\mathcal{C}$  be an  $[n, k, d]$ -linear code with  $h$ -level hierarchical locality having locality parameters  $[(r_1, \delta_1), (r_2, \delta_2), \dots, (r_h, \delta_h)]$ . Then

$$d \leq n - k + 1 - \sum_{\ell=1}^{\ell=h-1} \left( \left( \left\lceil \frac{k}{r_\ell} \right\rceil - 1 \right) (\delta_\ell - \delta_{\ell+1}) \right) - \left( \left\lceil \frac{k}{r_h} \right\rceil - 1 \right) (\delta_h - 1). \quad (14)$$

*Proof:* The proof is a straightforward extension of that of Thm. 2.1. The algorithm 2, (see flow chart in Fig. 5(b)) identifies a  $(k-1)$ -dimensional punctured code  $\mathcal{C}_s$  of  $\mathcal{C}$ , having large support. Then we will use the Fact in (2). The algorithm identifies a level-1 code that accumulates rank, and subsequently visits a level-2 code from within that, and continues recursively upto reaching a level- $(h-1)$  code that accumulates rank. Then it picks up all the level- $h$  codes that accumulate rank. If no more level- $h$  codes can be picked up, it steps back one level up, and finds a new level- $(h-1)$  code that accumulates rank. This can be viewed as a depth-first search for rank-accumulating level- $h$  codes. At each level, incremental support is added to the variable  $\Psi$ . Vaguely speaking, the incremental support that is added at each level depends on the minimum distance of the code at that level. Let  $a_{i_h}$  denote the incremental rank and  $s_{i_h}$  denote the incremental support while adding a level- $h$  code

---

#### Algorithm 2 For the proof of Thm. 3.1

---

```

1: Let  $i_1 = 0, i_2 = 0, \dots, i_h = 0; \Psi = \Phi, \ell = 1;$ 
2:  $M_0 = \mathcal{C}, M_1 = \Phi, M_2 = \Phi, \dots, M_h = \Phi$ 
3: while ( $\text{rank}(G|_\Psi) < k$ ) do
4:   if ( $\exists$  a level- $\ell$  code  $L_{i_\ell} \in M_{\ell-1}$  such that  $\text{rank}(G|_{\Psi \cup \text{Supp}(L_{i_\ell})}) > \text{rank}(G|_\Psi)$ ) then
5:      $M_\ell = L_{i_\ell}$ 
6:     if ( $\ell$  equals  $h$ ) then
7:        $\Psi = \Psi \cup \text{Supp}(M_h)$ 
8:        $S_{\text{last}} = \text{Supp}(M_h)$ 
9:        $i_h = i_h + 1$ 
10:    else
11:       $\ell = \ell + 1$ 
12:    end if
13:  else
14:     $\ell = \ell - 1;$ 
15:     $\Psi = (\Psi \setminus S_{\text{last}}) \cup T_\ell$ 
16:     $S_{\text{last}} = T_\ell$ 
17:     $i_\ell = i_\ell + 1$ 
18:  end if
19: end while

```

---

$L_{i_h}$ . By the algorithm,  $a_{i_h} > 0$ . The set  $S_{i_h}$  denotes the support of the level- $h$  code  $L_{i_h}$ . The set  $T_\ell$  denotes the incremental support of the level- $\ell$  code  $M_\ell$ , along with the columns from the last code that accumulated rank. It will contain at least  $(d_\ell - 1)$  columns in addition to the incremental rank. Let  $i_{\ell_{\text{end}}}$  denote the final value of the variables  $i_\ell, 1 \leq \ell \leq h$  before the algorithm terminates. Since  $a_{i_h} > 0$ , we can get non-trivial lower bounds on  $s_{i_h}$  and  $t_\ell := |T_\ell|$  quite similar to the proof of

Thm. 2.1. The rank is accumulated to  $k$  after adding the last local code  $L_{i_{h_{\text{end}}}}$ . By this time, we would have also visited  $i_{\ell_{\text{end}}}$  level- $\ell$  codes. Hence clearly,

$$i_{\ell_{\text{end}}} \geq \left\lceil \frac{k}{r_\ell} \right\rceil, \quad 1 \leq \ell \leq h$$

After adding  $i_{h_{\text{end}}} - 1$  level- $h$  codes, we would have accumulated rank that is less than or equal to  $(k - 1)$ . Hence we can always pick  $(k - 1) - \sum_{i_h=1}^{i_{h_{\text{end}}}-1} a_{i_h}$  columns from  $L_{i_{h_{\text{end}}}}$  so that the total rank accumulated becomes  $(k - 1)$ . The resultant punctured code is identified as  $\mathcal{C}_s$ . Following a similar line of arguments as in the proof of Thm. 2.1, we can get an estimate on  $\text{Supp}(\mathcal{C}_s)$  as

$$\begin{aligned} |S| \geq & (k - 1) + \sum_{\ell=1}^{\ell=h-1} \left( \left( \left\lceil \frac{k}{r_\ell} \right\rceil - 1 \right) (d_\ell - d_{\ell+1}) \right) \\ & + \left( \left\lceil \frac{k}{r_h} \right\rceil - 1 \right) (d_h - 1). \end{aligned}$$

Hence the theorem follows. ■

### B. Code Construction For All-Symbol Locality

The construction in Sec. II-D of the main text can be generalized to construct codes with  $h$ -level hierarchical locality containing  $[n_i, r_i, \delta_i]$  codes as  $i$ -th level code for each  $i = 1, 2, \dots, h$ . Here also, we require to satisfy a divisibility condition  $n_h \mid n_{h-1} \mid \dots \mid n$ . The generalization is straightforward, and it boils down to finding a finite field  $\mathbb{F}_{p^m}$  such that  $n_h \mid n_{h-1} \mid \dots \mid n_1 \mid (p^m - 1)$ , and  $n \leq (p^m - 1)$ . Then we can find a subgroup chain  $H_h \subset H_{h-1} \subset \dots \subset H_0 = \mathbb{F}_{p^m}^*$ . This allows us to create a coset-tree of depth  $h$ , and the code construction follows naturally. It can also be proved that the construction thus obtained will be optimal in terms of minimum distance if either of the two conditions holds:

- 1)  $r_h \mid r_2 \mid \dots \mid r_1 \mid k$ .
- 2)  $d = n_h + \delta_h$ ,  $\frac{n}{n_h} = \left\lceil \frac{k}{r_h} \right\rceil + 1$ ,  $\frac{n}{n_i} = \left\lceil \frac{k}{r_i} \right\rceil$ ,  $\forall i = 1, 2, \dots, h - 1$ .

### REFERENCES

- [1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.
- [2] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the Locality of Codeword Symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6925–6934, Nov. 2012.
- [3] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," in *Proc. IEEE Int. Symp. Network Computing and Applications (NCA)*, 2007, pp. 79–86.
- [4] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration and erasure correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4637–4660, Jun. 2014.
- [5] N. Silberstein, A. S. Rawat, and S. Vishwanath, "Error resilience in distributed storage via rank-metric codes," in *Allerton Conference on Commun., Control, and Computing, USA*, Oct. 2012, pp. 1150–1157.
- [6] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," in *Proc. IEEE Int. Symp. Inform. Theory, Istanbul, Turkey*, Jul. 2013, pp. 1814–1818.
- [7] N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with locality for two erasures," pp. 1962–1966, Jul. 2014.
- [8] A. Wang and Z. Zhang, "An integer programming based bound for locally repairable codes," *CoRR*, vol. abs/1409.0952, 2014.
- [9] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, pp. 4661–4676, May 2014.
- [10] S. Goparaju and A. R. Calderbank, "Binary cyclic codes that are locally repairable," in *Proc. IEEE Int. Symp. Inform. Theory, Honolulu, HI, USA*, Jul. 2014, pp. 676–680.
- [11] N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath, "Optimal locally repairable codes via rank-metric codes," in *Proc. IEEE Int. Symp. on Inform. Theory, Istanbul, Turkey*, Jul. 2013, pp. 1819–1823.
- [12] Z. Huang, E. Biersack, and Y. Peng, "Reducing repair traffic in P2P backup systems: Exact regenerating codes on hierarchical codes," *ACM Trans. Storage*, vol. 7, no. 3, pp. 10:1–10:41, Oct. 2011.
- [13] A. Wang and Z. Zhang, "Repair locality with multiple erasure tolerance," *IEEE Trans. Inf. Theory*, vol. 60, no. 11, pp. 6979–6987, Aug. 2014.
- [14] L. Parnies-Juarez, H. D. L. Hollmann, and F. E. Oggier, "Locally repairable codes with multiple repair alternatives," in *Proc. IEEE Int. Symp. on Inform. Theory, Istanbul, Turkey*, Jul. 2013, pp. 892–896.
- [15] A. Duminuco and E. Biersack, "A practical study of regenerating codes for peer-to-peer backup systems," in *Proc. IEEE Int. Conference on Distributed Computing Systems*, 2009, pp. 376–384.

### APPENDIX A

#### FLOWCHARTS FOR ALGORITHMS USED TO DERIVE BOUNDS

The flow charts for algorithms 1 and 2 are shown in Fig. 5(a) and Fig. 5(b) respectively.

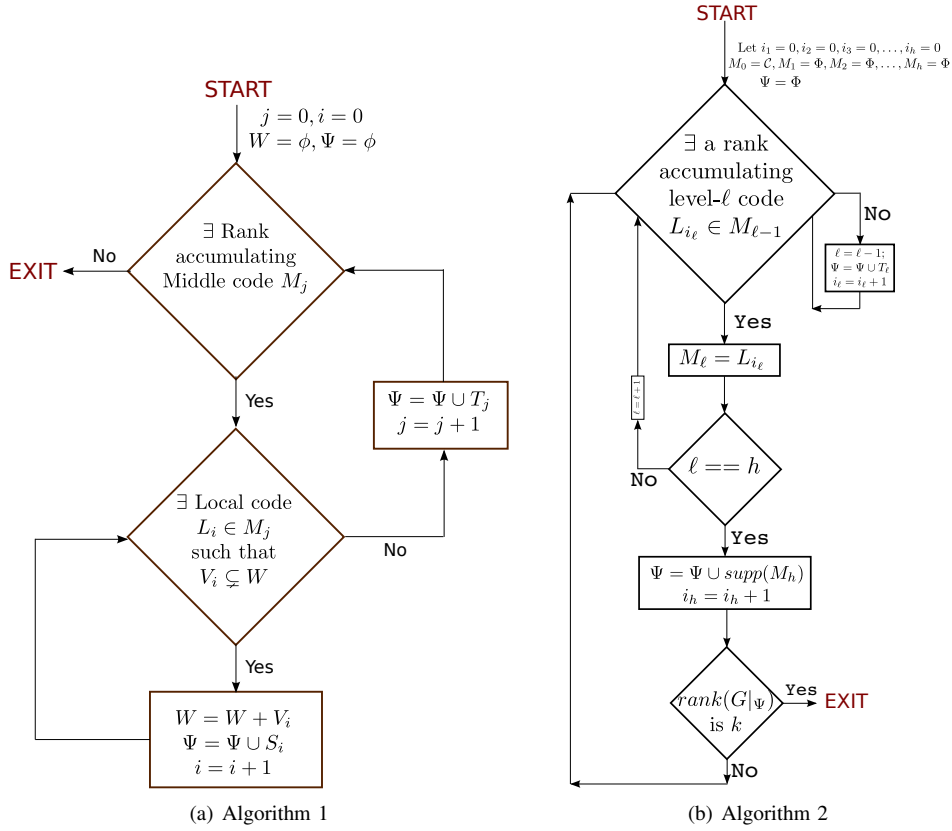


Fig. 5: Flowchart for algorithms uses for proving the bounds on minimum distance.

## APPENDIX B EXISTENCE OF REQUIRED FIELD

First we will show that there exists a prime  $p$  such that  $n_1 \mid p - 1$ . By Dirichlets theorem, if  $a$  and  $d$  are two co-prime numbers, then the sequence  $a, a + d, a + 2d, \dots$  will contain infinitely many primes. By setting  $a = n_1 + 1$  and  $d = n_1$ , we observe that there are infinitely many primes of the form  $(n_1 + 1) + \ell n_1$ , i.e. of the form  $(\ell + 1)n_1 + 1$ . Thus we obtain a prime  $p$  such that  $n_1 \mid p - 1$ . If  $n \leq p - 1$ , we are done. If not, pick a sufficiently large  $m$  such that  $n < p^m$ . Since  $n_1 \mid p - 1$ , we must also have  $n_1 \mid p^m - 1$ .

## APPENDIX C PROOF OF LEMMA 2.2

It is sufficient to verify that

$$q_{(i,\underline{t})}(X) \mid E_{(i,\underline{t})}(X) \quad (15)$$

$$\sum_{\substack{\mu_i \\ t_i=1 \\ t_j \text{ fixed for } j \neq i}} E_{(i,\underline{t})}(X) = 1 \bmod (X^{n_{i+1}} - \gamma_{(i-1,\underline{\tau})}^{n_{i-1}}). \quad (16)$$

where  $\underline{\tau}$  is the unique element such that  $(i-1, \underline{\tau}) = \pi(i, \underline{t})$  for every  $\underline{t}$  participating in the summation. For every such  $\underline{t}$ , the roots of  $q_{(i,\underline{t})}(X)$  are precisely

$$\Lambda_{(i,\underline{t})} = \gamma_{(i-1,\underline{\tau})} H_{i-1} \setminus A_{(i,\underline{t})}. \quad (17)$$

It can also be checked that  $E_{(i,\underline{t})}(X)$  evaluates to zero at any point in  $\Lambda_{(i,\underline{t})}$ . Hence  $q_{(i,\underline{t})}(X) \mid E_{(i,\underline{t})}(X)$ . It can also be seen that at any point  $y \in \gamma_{(i-1,\underline{\tau})} H_{i-1}$ , all except one term in the L.H.S. of (16) evaluates to zero, and the remaining term evaluates to 1. Hence (16) holds, thereby completing the proof.